UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/047,801	01/15/2002	John A. Cook	AUS920010995US1	4753

35525   7590   05/23/2005

IBM CORP (YA)
C/O YEE & ASSOCIATES PC
P.O. BOX 802333
DALLAS, TX 75380

EXAMINER
BULLOCK JR, LEWIS ALEXANDER

ART UNIT	PAPER NUMBER
2195	

DATE MAILED: 05/23/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)
	10/047,801	COOK, JOHN A.
	Examiner	Art Unit
	Lewis A. Bullock, Jr.	2195

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE _3_ MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
  Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on _18 February 2005_.
2a)☐ This action is **FINAL**.   2b)☒ This action is non-final.
3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) _1-3,5,7,9-14,16,18 and 20-30_ is/are pending in the application.
    4a) Of the above claim(s) _____ is/are withdrawn from consideration.
5)☐ Claim(s) _____ is/are allowed.
6)☒ Claim(s) _1-3,5,7,9-14,16,18 and 20-30_ is/are rejected.
7)☐ Claim(s) _____ is/are objected to.
8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.
10)☒ The drawing(s) filed on _19 March 2002_ is/are: a)☒ accepted or b)☐ objected to by the Examiner.
    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
    a)☐ All   b)☐ Some * c)☐ None of:
      1.☐ Certified copies of the priority documents have been received.
      2.☐ Certified copies of the priority documents have been received in Application No. _____.
      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1)☒ Notice of References Cited (PTO-892)
2)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3)☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.

4)☐ Interview Summary (PTO-413) Paper No(s)/Mail Date _____.
5)☐ Notice of Informal Patent Application (PTO-152)
6)☐ Other: _____.

DETAILED ACTION

Claim Rejections - 35 USC § 101

1.   35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of
> matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the
> conditions and requirements of this title.

Claims 1-3, 5, 7, 9-11, 22-30 are rejected under 35 U.S.C. 101 because the claimed

invention is directed to non-statutory subject matter. The cited claims detail a method /

apparatus / computer program product for creating a software state machine. The

method and apparatus claims do not recite any tangible structure and therefore are non-

statutory as detailed in the M.P.E.P. 2106. The computer program product claims are

signals as disclosed in the specification at page 24, lines 5-8 and therefor are also not

tangible as disclosed in the M.P.E.P. 2106.

Claim Rejections - 35 USC § 103

2.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set
> forth in section 102 of this title, if the differences between the subject matter sought to be patented and
> the prior art are such that the subject matter as a whole would have been obvious at the time the
> invention was made to a person having ordinary skill in the art to which said subject matter pertains.
> Patentability shall not be negatived by the manner in which the invention was made.

3.   Claims 1-3, 5, 7, 9-14, 16, 18, 20, 21-26, 28 and 29 are rejected under 35 U.S.C.

103(a) as being unpatentable over "The Reflective State Pattern" by FERREIRA et al.

As to claim 1, FERREIRA teaches a method for creating a software state

machine, comprising: providing a state machine object (client / context object of the

application level) (pg. 8, fig. 4); and providing an initializer object (FSMController object)

(pg. 8, fig. 4), wherein the initializer object defines states, actions, and conditions for a

software state machine (pg. 4, fig. 2, "create FSMStates()", "create FSMTransitions();

pg. 5, "FSMController...configures the FSM level, instantiating and initializing the

concrete FSMState and FSMTransition subclasses,"), wherein the state machine object

is configured to use the initializer object (pg. 3, "The FSM-level classes are responsible

for implementing the control aspects of the finite state machine, separating them from

the functional aspects that are implementd by the Context class and State classes at

the application level."; pg. 4, fig 2 wherein the client / context object invokes the

FSMController and the other meta-objects) to create a table object

(FSMConcreteState), wherein the state machine object (client / context object) is

configured to create state variables (state diagram specification), wherein the table

object (FSMConcreteState) is configured to create an array of state transition objects

(Transition meta-objects) based on the state variables (pg. 7, "The configuration of the

FSM level consist of: instantiation of each concrete subclass of the FSMState class and

FSMTransition class, initialization of the FSMState metaobjects with their corresponding

FSMTransitions, metaobject; initialization of each FSMTransition metaobject with its

corresponding next FSMState metaobject. The FSMController metaobject is

responsible for implementing all these configurations according to the state diargram's

specification of a Context class."; pg. 8, "A FSMConcreteState can have a list of

FSMTransitions, and it can delegate the handling of the transition sequentially or

concurrently."; pg. 10) and return the array transition objects (Transitions objects) to the

state machine object (via returning the results to the client, hence if the service was an instantiation request, either a normal return which would indicate creation, or error) (pg. 8; pg. 10), and wherein the state machine object is configured to execute the software state machine using the array of state transition objects (via making service requests to the FSMController to be handled by the Transition objects) (pg. 7-8, in particular fig. 4). However, FERREIRA does not explicitly teach that the variables are an array of variables. FERREIRA does not that the context object sends an state diagram specification that is used to create the state objects and transition objects. It would have been obvious to one of ordinary skill in the art that this specification would have to have a plurality of information to know how to instantiate the cited meta-objects and therefore would obviously be stored in an array since an array is well known to contain a plurality of data.

As to claims 2 and 3, FERREIRA substantially discloses the invention above. However, FERREIRA does not teach that the state machine object has an object constructor method. Official Notice is taken in that it is well known in the art that a state machine has a constructor method, i.e. an initialization method that it executes when the object is created and therefore would be obvious in view of FERREIRA that the constructor method executes to initialize its sub-objects to be used.

As to claim 5, FERREIRA teaches the initializer object (FSMController) includes a table element array creation method (createFSMState) and wherein the state machine

object (context object) is configured to call the table of element array creation method to

create the table object using the results of the table element array creation method (pg.

4, fig. 2, pg. 5, FSMState, "..Defines a method that initializes itself with a list of

FSMTransactions.."; pg. 5, FSMController, "...Configures the FSM level, instantiating

and initializing the concrete FSMState and FSMTransition subclasses...").

As to claim 7, FERREIRA teaches the initializer object (FSMController) includes

a table variable array creation method (createFSMState) and wherein the state machine

object is configured to call the table variable array creation method to create the array of

state variables (instantiated meta-objects according to the state diagram specification)

using the results of the table variable creation method (pg. 4, fig. 2, pg. 5, FSMState,

"..Defines a method that initializes itself with a list of FSMTransactions.."; pg. 5,

FSMController, "...Configures the FSM level, instantiating and initializing the concrete

FSMState and FSMTransition subclasses...")..

As to claim 9, FERREIRA teaches at least one of the state machine object

(FSMController) and the initializer object (context object / client) implements an

interface (set of functions) (pg. 4, figure 2; pg. 5-6).

As to claim 10, FERREIRA teaches the state machine object (FSMController)

includes a state method that is configured to return a current state of the software state

machine (pg. 5, FSMController, "Maintains the reference to the FSMState metaobject

that represents the current state, and delegates to it the handling of the intercepted

messages.").

As to claim 11, FERREIRA teaches a method for creating software state

machines, comprising: provide a state machine object (client / context object of the

application level) (pg. 8, fig. 4); providing an initializer object (FSMController object) (pg.

8, fig. 4), wherein the initializer object defines a plurality of states, one or more actions,

one or more inputs, one or more conditions, one or more events, and one or more

triggers for a software state machine (pg. 4, fig. 2, "create FSMStates()", "create

FSMTransitions(); pg. 5, "FSMController...configures the FSM level, instantiating and

initializing the concrete FSMState and FSMTransition subclasses,"), wherein the state

machine object is configured to use the initializer object (pg. 3, "The FSM-level classes

are responsible for implementing the control aspects of the finite state machine,

separating them from the functional aspects that are implemented by the Context class

and State classes at the application level."; pg. 4, fig 2 wherein the client / context object

invokes the FSMController and the other meta-objects) to create a table object

(FSMConcreteState), to create an array of state transition objects (Transition meta-

objects) (pg. 7, "The configuration of the FSM level consist of: instantiation of each

concrete subclass of the FSMState class and FSMTransition class, initialization of the

FSMState metaobjects with their corresponding FSMTransitions, metaobject;

initialization of each FSMTransition metaobject with its corresponding next FSMState

metaobject. The FSMController metaobject is responsible for implementing all these

configurations according to the state diargram's specification of a Context class."; pg. 8,

"A FSMConcreteState can have a list of FSMTransitions, and it can delegate the

handling of the transition sequentially or concurrently."; pg. 10), wherein each state

transition object in the array of state transition objects defines at least one of the one or

more conditions (guard-conditions) that causes a given state transition in the software

state machine (pg. 5, "Each FSMTransition has information about the transition function

(the event, the guard-conditions and the exit action) that should be verified unless a

transition is triggered."), and responsive to occurrence of a trigger, evaluating the one or

more inputs (the event), computing the one or more conditions (guard-conditions), and

determining a next state (via performing the exit action which tells the FSMController to

change the state) based on the array of state transition objects (pg. 5). However,

FERREIRA does not explicitly teach that the condition is a boolean condition.

FERREIRA does teach that if the event triggers transition is true, then to invoke a

change state method to change the state (pg. 8, figure 4). Therefore it would be

obvious to one skilled in the art at the time of the invention that the conditions are

Boolean conditions since, Boolean conditions typically return either true or false.

As to claim 25, FERREIRA teaches a given state transition object (transition

object) in the array of state transition objects defines an action (change state) to take

responsive to a given state transition (pg. 8, figure 4).

As to claim 26, FERREIRA teaches a given state transition object in the array of state transition objects defines an event (change state message) to be generated responsive to a given state transition (pg. 8, figure 4).

As to claim 23, reference is made to a program product that corresponds to the method of claim 1 and is therefore met by the rejection of claim 1 above.

As to claims 24, 28 and 29, reference is made to a program product that corresponds to the method of claims 11, 25 and 26 and is therefore met by the rejection of claims 11, 25 and 26 above.

As to claims 12-14, 16, 18, 20 and 22, reference is made to an apparatus that corresponds to the method of claims 1-3, 5, 7, 9 and 10 and is therefore met by the rejection of claims 1-3, 5, 7, 9, and 10. Claim 12 further details a processor and a memory storing the objects. Official Notice is taken in that it is well known in the art that a computer system having a processor and memory can execute an application. FERREIRA teaches the invocation among components of an application (pg. 1). Therefore, it would be obvious to one skilled in the art at the time of the invention that the invocation of the program is performed on a computer system having a processor and memory.

As to claim 22, refer to claim 1 for rejection. However, claim 22 further details a virtual machine having the instance of the state machine object, the state machine object having a object constructor method that creates the initializer object, and a state array creation method to create the state transition objects to execute the state machine. FERREIRA teaches a state array creation method (createFSMState) and wherein the state machine object (context object) is configured to call the table of state array creation method to create the table object using the results of the state array creation method (pg. 4, fig. 2, pg. 5, FSMState, "..Defines a method that initializes itself with a list of FSMTransactions.."; pg. 5, FSMController, "...Configures the FSM level, instantiating and initializing the concrete FSMState and FSMTransition subclasses..."). FERREIRA also teaches the invocation among components of an application (pg. 1). However, FERREIRA does not teach a virtual machine having the objects. Official Notice is taken in that it is well known in the art that a virtual machine instantiates and executes objects. Therefore, it would be obvious to one skilled in the art at the time of the invention that the invocation of the program is performed on a computer system having a virtual machine, processor and memory. In addition, FERRIERA does not teach that the constructor method.

Official Notice is taken in that it is well known in the art that a state machine has a constructor method, i.e. an initialization method that it executes when the object is created and therefore would be obvious in view of FERREIRA that the constructor method executes to initialize its sub-objects, i.e. initializer object, to be used.

4.   Claims 27 and 30 are rejected under 35 U.S.C. 103(a) as being unpatentable

over "The Reflective State Pattern" by FERREIRA et al. in view of NGUYEN (U.S.

Patent 6,751,753).

As to claim 27, FERRIERA substantially discloses the invention above.

However, FERRIERA does not teach the graphical user interface. NGUYEN teaches

presenting one or more graphical user interfaces for defining the software state machine

(col. 9, lines 39-55). It would be obvious that since the software state machine

comprises of states, actions, inputs, and conditions and the graphical user interfaces

allows a user to specify state machine modifications that the modifications would entail

states, actions, inputs, and conditions. Therefore, it would be obvious to one skilled in

the art at the time of the invention to combine the teachings of FERIERA with the

teachings of NGUYEN in order to access and configure a state machine.

As to claim 30, reference is made to a program product that corresponds to the

method of claim 27 and is therefore met by the rejection of claim 27 above.

Relevant Prior Art Cited, but not Relied Upon

"On the Implementation of Finite State Machines" by Van Gurp et al. and "State-

Oriented programming" by SAMEK teaches the initialization of a finite state machine via

an initialization method / constructor that initializes the remaining state and transition

objects.

U.S. Patent Publication 2002/0144015 A1, herein Lortz, teaches the creation of a state machine via a factory object that creates the necessary state and transition map for the state machine.

Response to Arguments

5.   Applicant's arguments with respect to claims 1-3, 5, 7, 9-14, 16, 18 and 20-30 have been considered but are moot in view of the new ground(s) of rejection.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Lewis A. Bullock, Jr. whose telephone number is (571) 272-3759. The examiner can normally be reached on Monday-Friday, 8:30 - 5:00.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Meng An can be reached on (571) 272-3756. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

May 12, 2005

LEWIS A. BULLOCK, JR.
PRIMARY EXAMINER